LA-UR-21-29114
Approved for public release; distribution is unlimited.

Title: DRiFT - RELEASE 1.0.0 ORGANIC SCINTILLATORS

Author(s): Andrews, Madison Theresa
Bates, Cameron Russell
Mckigney, Edward Allen
Mullen, Austin Daniel
Woldegiorgis, Surafel Fantaye
Rising, Michael Evan
Marcath, Matthew James
Sood, Avneet

Intended for: DRiFT Manual to be released with Exectuable

Issued: 2021-09-15

DRiFT - RELEASE 1.0.0
ORGANIC SCINTILLATORS

DRiFT CONTRIBUTORS:

MADISON ANDREWS*[1], CAMERON BATES[1], EDWARD MCKIGNEY[1]
AUSTIN MULLEN[1], SURAFEL WOLDEGIORGIS[1], MICHAEL RISING[2],
MATTHEW MARCATH[1], AND AVNEET SOOD[1]

[1]*XCP-7: Radiation Transport Applications*
[2]*XCP-3: Monte Carlo Codes*
X-Computational Physics Division
Los Alamos National Laboratory
**madison@lanl.gov*

LAST UPDATED: SEPTEMBER 3, 2021
LOS ALAMOS NATIONAL LABORATORY TECHNICAL REPORT
LA-UR-21-XXXXX

Notice

Authors: Madison Andrews, Cameron Bates, Edward McKigney, Austin Mullen, Surafel Woldegiorgis, Avneet Sood, Matthew Marcath

Abstract

DRiFT (a Detector Response Function Toolkit) is LANL-developed software that post-processes output from the extensively validated radiation transport code, MCNP [1], and generates realistic nuclear instrumentation response. DRiFT is designed to be flexible, enabling users to specify detector type and many experimental settings, as well as accommodating the addition of their own desired features. Although DRiFT development has included scintillator [2], gas [3], and semiconductor features [4], the focus of this release is on organic scintillator and associated capabilities. Organic scintillators are widely used in the areas of nuclear safeguards and nuclear non-proliferation efforts [5, 6]. DRiFT has several diagnostic and detector physics features relevant to detailed scintillator simulations including: tracking source particle information, scintillation light production, the effects of PMT quantum efficiency and gain, and digitizer settings. Users can select responses from many scintillator and PMT types supported natively by DRiFT, or add their own by following the instructions in this document. We acknowledge that DRiFT is under active development, bug reports and general questions and comments should be directed to Madison Andrews, madison@lanl.gov.

This manual is divided into four parts: I) An overview of DRiFT, including how to obtain and install the executable, II) A description of the detector physics related to scintillators available, III) a description of more general DRiFT features the user may find useful, and IV) a description of the test suite and examples made available with the code release.

Acknowledgements

We acknowledge the funding provided by the Los Alamos National Laboratory Directed Research and Development (LDRD) Program, the Office of Nuclear Verification, NNSA, and US DOE. Additionally, scintillation measurement data provided by K. Meierbachtol and T. Borgwardt is much appreciated. We also appreciate the feedback from our first tester, David Broughton. Finally, the 2021 release of organic scintillator capabilities and documentation was made possible with Laboratory Technology Evaluation and Demonstration Funding.

Acronyms

Table 1: Acronyms Contained in DRiFT Manual

DOE	Department of Energy
DRiFT	Detector Response Function Toolkit
EJ	ElJen Technology
FCI	Feynman Center for Innovation
LANL	Los Alamos National Laboratory
LDRD	Laboratory Directed Research and Development
MCNP	Monte Carlo N-Particle
NNSA	National Nuclear Security Administration
PE	Photon Emission
PMT	PhotoMultlipier Tube
PSD	Pulse Shape Discrimination
PTRAC	Particle TRAcking File
QE	Quantum Efficiency
RSICC	Radiation Safety Information Computational Center
TED	Technology Evaluation and Demonstration
US	United States
XCP	X-Computational Physics

Contents

List of Figures

List of Tables

Part I

DRiFT Overview

Chapter 1

Obtaining DRiFT

DRiFT [2, 7, 8, 9, 10] will be released to approved users by the Feynman Center for Innovation (FCI). If you are interested in using DRiFT, please email Madison (madison@lanl.gov) who will forward your request to the FCI for approval. FCI will reach out to you or your institution with some follow up questions before issuing approval to use DRIFT. Once approved, the DRiFT executable can be sent by Madison directory. The DRiFT executable requires MCNP6.2 PTRAC files to post-process. MCNP is distributed by the Radiation Safety Information Computational Center (RSICC) website.

Chapter 2

Installing

This chapter is concerned with those obtaining a DRiFT executable. If you would like instructions on how to build DRiFT on LANL HPC systems, contact DRiFT developers for associated documentation. Currently DRiFT has only been developed for Linux operating systems.

Once you receive the DRiFT executable (drift_executable_only.tar.gz):

```
tar -xzvf drift_executable_only.tar.gz
```

The remainder of these instructions are also found in the README accompanying the DRiFT executable. To install DRiFT, execute the following commands (replacing the X's with your current version number):

```
./drift-X.X.X-Linux.sh
```

You may be prompted to choose whether the install will be placed in a sub-directory. If you are running the installer from a dedicated DRiFT directory, select no. If you are running the install from a more general directory (such as your home directory or a general software directory), select yes.

If you create a subdirectory, navigate into that directory:

```
cd drift-X.X.X-Linux
```

Run the install script:

```
source install_drift.sh
```

This script will conclude by running a series of unit tests. If all tests are passed, your DRiFT installation was successful and you are ready to use DRiFT!

The manual gives a good overview of the code, and there are a number of test cases that can help with through running DRiFT (Cf252_EJ301 test suite is recommended as a starting point).

Chapter 3

DRiFT Configuration Sections

3.1 List and Summary of Modules Available

Different modules simulate detector response, PMT and digitizer electronics in addition to diagnostic capabilities are listed with a short summary below. They are in the recommended order of use. They are described in more detail in other chapters and a list of the keywords can be found in the next section.

- `global` This is a (required) administrative module for `DRiFT` where your `MCNP` output and detector cells are defined.
- `SourceInformation` If you have included source particle information in your `MCNP` output, you can print out source particle information for each event.
- `TimeDistribution` Here you can define your source activity and `DRiFT` will sort the PTRAC events accordingly. This generates a realistic estimate of pulse-up and pulse distortions.
- `Scintillation` This is where you define the properties of your scintillator and PMT configuration, either using pre-defined libraries, or having `DRiFT` read in one specific to your set up.
- `Digitizer` Digitization effects are simulated with this module which can generate waveforms for each pulse. Note that the inputs required for requesting output showing the waveforms is provided under `WriteOutput`.
- `WriteOutput` In this module you can write out the information generated for each event which can include energy deposited, source particle information, waveforms and much more.

3.2 Configuration Sections, Keywords, and Options

Table 3.1: DRiFT Sections in Suggested Configuration File Order - global

Name	Keywords	Options
[global]		
	modeltype	event
	datafile	*ptrac filename*
	ptrac	bin
	num_det_cells	*integer, i.e. 1*
	det_cells	*integer, MCNP cell nums, separated by spaces*
	random_seed	*integer, i.e. 12321, defaults to system time*

Table 3.2: DRiFT Sections Keyword Options - SourceInformation

Name	Keywords	Options
[SourceInformation]		
	call	SourceInformation
	multi_src	no *(default)*
		yes
	source_particles_list	*integer, 1 (default corresponding to neutron)*
		1 2 3 ... (following MCNP convention.)
	print_src_particle_events	no *(default)*
		yes
	print_src_for_each_det	no *(default)*
		yes

Table 3.3: DRiFT Sections Keyword Options - Time Distribution

Name	Keywords	Options
[TimeDistribution]		
	call	TimeDistribution
	activity	*integer (Bq)*

Table 3.4: DRiFT Sections Keyword Options - Scintillation

Name	Keywords	Options
[Scintillation]		
	call	Scintillation
	detector	*Scintillator name, ej. EJ301*
	optical_transport	*double, default 0.6*
	voltage	*double, 1500 V, PMT voltage*
	pmt_type	*PMT name, i.e. 9821B*
	max_energy	*double, 25.0 MeV default*
	gain	*double, default set by PMT voltage and model*
	scint_yield	*double, default set by scintillator type*
	PE_file	*filename of scintillator emission spectrum*
	QE_file	*filename of PMT quantum efficiency spectrum*
	light_file	*filename of scintillator light output table*
	pulse_shape_file	*filename of user-defined pulse shape*
	rise_time	*double, rise time of the scintillator (in ns) for pulse shape*
	decay_fast	*double, fast decay time constant (in ns) for pulse shape*
	decay_slow	*double, slow decay time constant (in ns) for pulse shape*
	fast_decay_weight	*double, relative weight of fast decay time constant*
	pulse_arrival_time	*double, default 15 ns*

Table 3.5: DRiFT Sections Keyword Options - Digitizer

Name	Keywords	Options
[Digitizer]		
	call	Digitizer
	digitizer_samples	*int, 512*
	resolution	*int, 16384 default*
	voltage_range	*double, 2.0 V default*
	ter_res	*double, 50.0 ohm default*
	DC_offset	*double, 0.1 % default*
	start_point	*double, 0.1 by default*
	trigger_ADC	*int, 100 by default*
	rate	*double, 500.e6 default (Hz)*
	s_gate	*double, 22 e-9 by default (22 ns)*
	l_gate	*double, 90e-9 by default (90 ns)*
	PSD	*string, no by default*
	pileup	*string, no by default*
	digitizer_type	*string, none specified by default*
	waveform_out	*string, no by default*
	waveform_file	*string, none by default*

Table 3.6: DRiFT Sections Keyword Options - WriteOutput

Name	Keywords	Options
[WriteOutput]		
	call	WriteOutput
	num_outputs	*integer less than 10*
	outputs	source_e *(default), MeV*
		source_t *seconds*
		source_cell
		source_type
		count
		det_pulse
		det_cell
		corr_count
		PSD

Part II

Detector Physics - Scintillators

Chapter 4

The Scintillator Module

4.1 Overview

The scintillation module is intended to convert energy deposited by a proton, deuteron, or electron (MeV) into the electron equivalent energy (MeVee) based on a detector's properties. Each particle is treated separately to properly determine the amplitude and shape of the resulting pulse. The PTRAC particle's electron equivalent energy (MeVee) is determined for that specific particle type and original energy using the quenching data specific to a detector. The scintillation photon yield is specific to the scintillator material specified by `DRiFT` input. At this time `DRiFT` does not perform light collection efficiency calculations, rather the value is specified by the user (i.e. 0.60 results in an average of 60 % of the light photons making it to the photocathode). Currently `DRiFT` assumes the scintillator is coupled to a PMT and will calculate the amount of signal lost to quantum efficiency effects from the photocathode. Finally in the `Scintillation` module, the photons making it through the photocathode are converted to electrons and multiplied using a PMT specific gain (or one specified by the user). At the conclusion of the `Scintillation` module the events have been converted into an electrical current (intended to be processed by the `Digitizer` module).

4.2 MCNP Simulations

Currently `DRiFT` requires `MCNP` particle tracking (PTRAC) files as inputs for scintillator simulations. There are several `MCNP` input examples in the test suite. An example PTRAC line to track neutron detection in a scintillator is as follows:

```
PTRAC File=BIN MAX=1e9 WRITE=ALL type=h,n event=bnk,src
```

In the above example the file specification is optional as it will defaulted to binary. Often the maximum number of events will need to be increased if source particle information is also desired. In this example all events concerning neutrons and protons are written. If source information or cell flags are not desired `DRiFT` output, you can use

cell filters to write out only events occurring in detector cells. The test suites, namely `CF252_EJ301` also display suggested `PHYS` card options for these simulations.

4.3 DRiFT Keywords, Options, and Descriptions

4.3.1 call

DRiFT requires that you set the call in the `Scintillation` module to Scintillation:

```
call=Scintillation
```

4.3.2 Scintillator Specifics

detector

DRiFT has many scintillator detector options supported natively, and the user can also add their own (see next chapter). Options in version 1.0.0 of the code for scintillator detectors are listed below, and an example is also shown below. Note, if your desired detector is not available, you can add your own response following instructions in the following chapter.

```
detector=EJ301
```

Scintillator options supported natively in `DRiFT` include: EJ-301, EJ-212, EJ-309, NE-102, NE-213, Stilbene, BaF2, EJ-228, Pilot U, U, EJ-230, Pilot U2, EJ-200, Pilot F, Stilbene-d, EJ-215, and NE-213.

scint_yield

By default `DRiFT` will determine the number of photons per MeVee based on the detector type specified by the user. You may override this with scint_yield keyword, although not recommended.

PE_file

If you have built your own scintillator, you must provide the name of the scintillator emission spectrum file. Please see the following chapter for more information..

light_file

If you have built your own scintillator, you must provide the name of the light output table file. Please see the following chapter for more details.

optical_transport

At this time `DRiFT` does not perform light collection efficiency calculations (i.e. how many of the produced photons make it to the photocathode) and therefore this value must be specified by the user. It should range from 0 - 1.0 and dictates the mean % of photons that reach the PMT interface.

max_energy

This option specifies the larges incident particle energy you would like a pulse shape for, it is recommended that you leave at the default value of 25.0 MeV.

4.3.3 PMT Specifics

pmt_type

`DRiFT` has many PMT options supported natively, and the user can also add their own (see next chapter). Options in version 1.0.0 of the code for PMT are: none, 9721B, 3177, XP2020, B133D)1, L133D29, 9202B, 9202QB, 9266B, 9266QB, 9265B, 9125B, 9125QB, and 9125WB. Note, if your desired detector is not available, you can add your own response following instructions in the following chapter.

voltage

PMT gains are dependent on the voltage. Many of PMT types supported include a list of voltage and corresponding gain. So the user can specify a voltage and `DRiFT` will determine the appropriate gain. To see what voltage options are available for natively supported PMTs, check under the gain sections of the appropriate datafile located in `data/pmt`.

gain

If you wish to specify the gain yourself, you can. This will overwrite any voltage dependent gain determined with the previous voltage keyword specification.

QE_file

If you have built your own PMT, you must provide a quantum efficiency file, see next section for details.

4.3.4 Pulse Shape Specifications

More details about defining your own detector pulse shape can be found in the next Chapter.

pulse_shape_file

In this option you can specify a pulse shape file `DRiFT` will read to determine an average pulse shape, please see next chapter for more details.

rise_time

When customizing your own pulse shape, you can define the rise time (in ns).

decay_fast

A double value (in ns) which defines the fast decay time constant for a pulse shape.

decay_slow

A double value (in ns) which defines the slow decay time constant for a pulse shape.

fast_decay_weight

A double value which defines the relative weight of the fast decay component.

Chapter 5

Adding Scintillators and PMTs for DRiFT Executable Use

Note, there are different instructions available for developers at LANL wishing to add a native feature to `DRiFT`.

5.1 Light Output Function

The electron equivalent energy for a scintillating material can be calculated from the recoil proton energy using an analytical function cited by Cecil et. al., see Eq. 5.1. The optimized fitting parameters cited by Hansen 2002 can be seen in Table 5.1. Different parameters for different scintillators can be found in the open literature, or results from a light output experiment can be fit to such a function to obtain your own parameters. These parameters are used in `DRiFT` as described in the sections below.

$$L = aE_p - b[1 - exp(-cE_p^d)] \tag{5.1}$$

Table 5.1: Optimization constants for stilbene light output function

Parameter	Value
a	0.693
b	3.0
c	0.2
d	0.965

5.2 Photon Emission Spectrum

The photon emission spectrum of a scintillator is generally available from the manufacturer's website or specification sheet. This information should be entered into a simple text file with the first column being the wavelength of light in nanometers and the second column being the relative light output at that wavelength, separated by a single space. The data does not need to be normalized. The reader is referred to the file data/scintillators/EJ301EmissionSpectrum for an example of a properly formatted data file.

5.3 PMT Quantum Efficiency

The quantum efficiency of a PMT at different wavelengths is generally available from the manufacturer's website or specification sheet. This information should be entered into a simple text file with the first column being the wavelength of light in nanometers and the second column being the quantum efficiency of the PMT at that wavelength, in units of percentage (i.e. 30% should be entered as 30, **NOT** 0.30). You may need to convert from radiant sensitivity to quantum efficiency, which can be done with the following equation, provided from the HZC Photonics Photomultiplier Tubes Manual:

$$QE(\%) = \frac{124}{\lambda(nm)} * radiant\ sensitivity(mA/W) \tag{5.2}$$

The reader is referred to the file data/pmt/QEBorosilicateGlass for an example of a properly formatted data file. The new quantum efficiency data file should be included in the directory data/pmt/.

5.4 PMT Gain and Other Information

In addition to the quantum efficiency data, an additional file needs to be added to the data/pmt directory. This data file must be named the same as the PMT as defined in the header files (i.e. for the "9821B" PMT, the file is named 9821B). In this file, you should include the number of dynodes, the dark rate, and the gain of the PMT. For listing the gain of the PMT, the first value provided should be the voltage, and the second value provided should be the gain at that voltage. An example of the data file is below:

```
[Misc]
n_dynodes = 12
dark_current = 500


[GainA]
2000=5e6
1500=4e5
```

5.5 Adding Scintillators and PMTs for `DRiFT` executable use

It is possible to use your own scintillators and PMTs in `DRiFT` without needing to modify or recompile the `DRiFT` code itself. To do this, you need to provide three data files for the code to model scintillator and PMT behavior:

- **PE_file**: the emission spectrum of the scintillator
- **light_file**: the light output response function of the detector to the charged particle of interest, in MeVee
- **QE_file**: the quantum efficiency of the PMT

Of these files, PE_file and QE_file have already been described in previous sections as they are always required for adding a new Scintillator or PMT, even if you also modify the header files. The light_file should contain a table of the light output response of the scintillator to a relevant charged particle (protons and deuterons are currently supported), where the particle energy is given in MeV and the light output is given in MeVee. An example of the formating of such a light file is available at `data/scintillators/lightfiles/light_output.EJ301.Verbinski`. These files can be given in the input file in one of three ways:

- As an absolute path.
- As a relative path from the DRIFTDATA environment variable.
- As a relative path from the file expected location. For PE_file, this is `DRIFTDATA/scintillators/`, for light_file this is `DRIFTDATA/scintillators/lightfiles/`, and for QE_file this is `DRIFTDATA/pmts/`.

In addition to these data files, you need to provide two additional input parameters in the `DRiFT` input file, both included under the `[Scintillation]` section. **scint_yield** should be the scintillation yield of the scintillator in photons per MeV. **gain** should be the gain of the PMT. Note that because the PMT is not available in the `DRiFT` code, you will not be able to define a voltage in order to automatically determine the PMT gain. An example of the `[Scintillation]` section with these inputs provided is given below.

```
[Scintillation]
call=Scintillation
detector=YourDetectorHere
optical_transport=0.6
pmt_type=YourPMTHere
gain=1e5
scint_yield=12e3
PE_file=EJ301EmissionSpectrum
QE_file=QEBorosilicateGlass
```

```
light_file=light_output.EJ301.Verbinski
```

An example case is given in the test suite, under `test_suite/Using_Data_Tables`, which gives a concrete example of how to impelement a new scintillator or PMT using this method.

5.6 Adding a Custom Pulse Shape

By default, DRiFT contains equations for calculating generic pulse shapes for electrons, neutrons, and deuterons. However, DRiFT can model a custom pulse shape in two different ways, as defined in the input parameter file. First, you can provide the rise time, fast decay constant, slow decay constant, and fast decay weighting factor in the input file, and DRiFT will compute a wavefunction based on the equation provided from F.Q.L Friesen [11]:

$$
\begin{aligned}
y(t) &= f(t)[Rg(t) + (1-R)h(t)] \\
f(t) &= (e^{-(t-t_0)/\tau_r} + 1)^{-1} \\
g(t) &= (e^{(t-t_0)/\tau_f} + 1)^{-1} \\
h(t) &= (e^{(t-t_0)/\tau_s} + 1)^{-1}
\end{aligned}
$$

Here, τ_r is the rise time, τ_f is the fast decay constant, τ_s is the slow decay constant, and R is the fast decay weighting factor, which must be between 0 and 1 (with 0 indicating no fast decay contribution, and 1 indicating all fast decay contribution). t_0 is the arrival time of the pulse, which can also be set by the user or left at a default 15 ns. An example of these parameters defined in an input file is below (all units are in nanoseconds):

```
[Scintillation]
call=Scintillation
...
rise_time=0.9
decay_fast=2.1
decay_slow=21.3
fast_decay_component=0.8
pulse_arrival_time=15
```

Alternatively, DRiFT can be given a text file containing a table that specifies the pulse shape. The table should contain two columns, the first being time in nanoseconds, and the second being the intensity of the pulse at that time. The intensity does not need to be normalized. If DRiFT needs to determine the intensity at a time between two defined points in the table, linear interpolation is used. If the pulse continues beyond the maximum extent of the table (i.e. the total digitizer acquisition is 250 ns long but the table only contains values up to 200 ns), DRiFT sets all intensity values beyond the maximum extent to zero and gives a warning. For an example of a pulse shape file, please see data/scintillators/sample_waveform. The pulse shape file can be called in the Scintillation field of the input file as follows:

```
[Scintillation]
call=Scintillation
...
pulse_shape_file=sample_waveform
```

Chapter 6

Digitizer

6.1 Overview

DRiFT's digitizer output is intended to simulate the raw waveforms collected during experiments. It converts the current output from the scintillator module into a digitized ADC value. These pulse integral is also converted into electron equivalent energy for those users who wish to simulate the effects of pulse digitization, but do not desire a ADC value as output.

6.2 Digitizer Keyword Options

Digitizer keywords which can be provided by the user are selected in Table 6.1. If these values are not defined in the configuration file DRiFT will display a warning in the terminal and default to the values displayed in Table 6.1. The majority of these values can found on the technical specification sheets provided by digitizer manufacturer.

6.2.1 Digitizer Settings and Specifications

Digitizer specifications can typically be found on the manufacturer's website. Options that can be set by the user in DRiFT include digitizer_samples which is the length of the event captured in the simulation, resolution which is how many ADC bits are available, so for example if you have a 14 bit digitizer the resolution specification would be 16384, the termination resistance ter_res which is used by DRiFT to convert the current from the Scintillation module into a corresponding voltage, and the voltage range of the digitizer which defaults to voltage_range=2.0. All of these settings affect the performance of the digitizer and can be utilized to test pulse shape discrimination algorithms, see next section for build in PSD options.

Table 6.1: DRiFT Sections Keyword Options - Digitizer

Name	Keywords	Options
[Digitizer]		
	call	Digitizer
	digitizer_samples	*int, 512*
	resolution	*int, 16384 default*
	voltage_range	*double, 2.0 V default*
	ter_res	*double, 50.0 ohm default*
	DC_offset	*double, 0.1 % default*
	start_point	*double, 0.1 by default*
	trigger_ADC	*int, 100 by default*
	rate	*double, 500.e6 (sampling frequency of 500 MHz) default*
	s_gate	*double, 22 e-9 (22 ns) by default*
	l_gate	*double, 90e-9 (90 ns) by default*
	PSD	*string, no by default*
	pileup	*string, no by default*
	digitizer_type	*string, none specified by default*
	waveform_out	*string, no by default*
	waveform_file	*string, none by default*

6.2.2 Pulse Shape Discrimination

Digitizer triggering is simulated by searching for the highest ADC value for each pulse. If this value exceeds the trigger ADC threshold, `trigger_ADC`, the event will be recorded as a pulse. The trigger start position is found, `start_point`, and used along with the short and long gates (`s_gate, l_gate` both in units of ns) to calculate a pulse shape discrimination (PSD) value if desired by the user. The digitizer ADC values are saved along with calculated PSD values for the user to access if desired. These PSD values can then be printed alongside energy to produce the plots seen in Figures 6.1, 6.2, and 6.3. Neutron events can be seen plotted in the top grouping of bins, while gamma events are separated into the bottom grouping.

6.2.3 Other Options

`pileup`

If the `pileup` is specified in the input, each PTRAC event (those originating from the same source particle) will be compared to subsequent pulses in that detector cell. If a following event is registered in the same detector with a time difference less than the length of the digitizer sample (which is dependent on `rate` and `samples`), the event is flagged as a possible pile up. Any pulse current value within the digitizer window is added to the previous event's current before the analog-to-digital conversion. Currently it is assumed that the dead-time of digitizer following a pulse is as long as the sampling

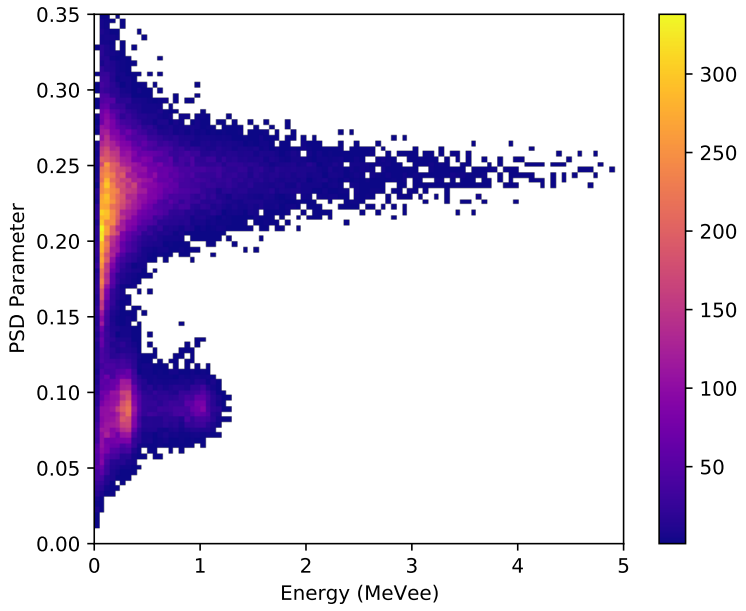

Figure 6.1: Histogram demonstrating the pulse shape discrimination capabilities of DRiFT using Cf252 neutrons and Co-60 gamma rays on an EJ301 detector.

length meaning that any pulses contributing to the preceding one would not be recorded separately.

6.2.4 Conversion to Energy

In the majority of cases it is expected the user will not want all of the details DRiFT is capable of simulating. If they wish for DRiFT to perform the energy calibration it does so by calculating the pulse integral of a 1 MeVee energy deposition in the detector volume. This pulse integral is used to calculate energy values for each simulated pulse which can be easily plotted using DRiFT's histogram option, or read from generated ASCII files.

6.2.5 Pulse Saturation

Reference [12] describes a simple test case used by DRiFT to assess code performance. In short, MCNP simulates neutron emissions from a Cf-252 source which create proton recoils in a detector volume filled with EJ-301 organic scintillator material. The protons are recorded and saved as a PTRAC output to be post-processed by DRiFT. DRiFT applies a detector response using PMT and digitizer settings corresponding to

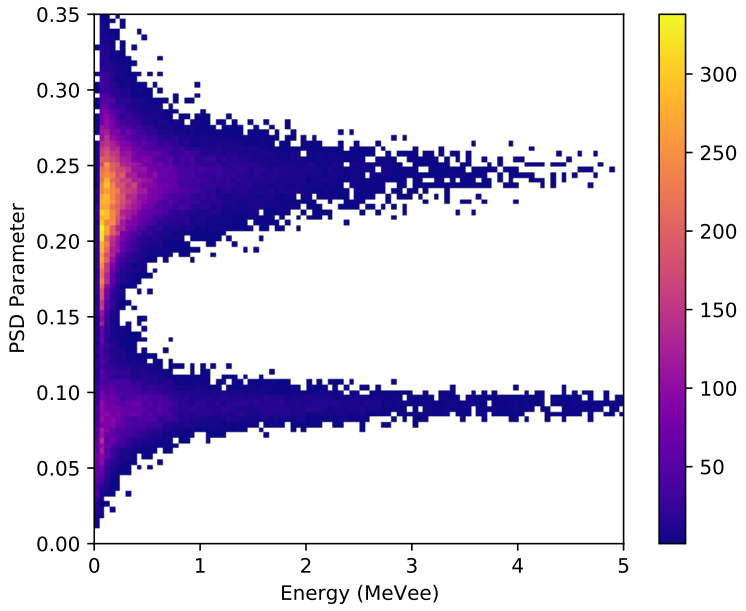

Figure 6.2: Histogram of DRiFT's ability to perform pulse shape discrimination on Cf252 neutrons and a gamma spectrum approximating the Cf252 prompt gammas.

those provided by the experimentalist. A comparison of DRiFT simulations with measurements is shown in Figure 6.4.

Figure 6.5 shows DRiFT output after an incorrect PMT voltage was used the simulation. The applied voltage was too high resulting in a larger PMT gain used when calculating the pulse current. When this was converted into an ADC value in DRiFT's digitizer class the pulses were saturated, creating the spectra shown in Figure 6.5. This example demonstrates an advantage of DRiFT when preparing for measurements. The users can try different digitizer and PMT settings and quickly discern if they are appropriate.

The user also has the option to generate the individual pulse shape from each detection event. See chapter on Text File Output for instruction on how to request a separate output file that contains the pulse shapes.

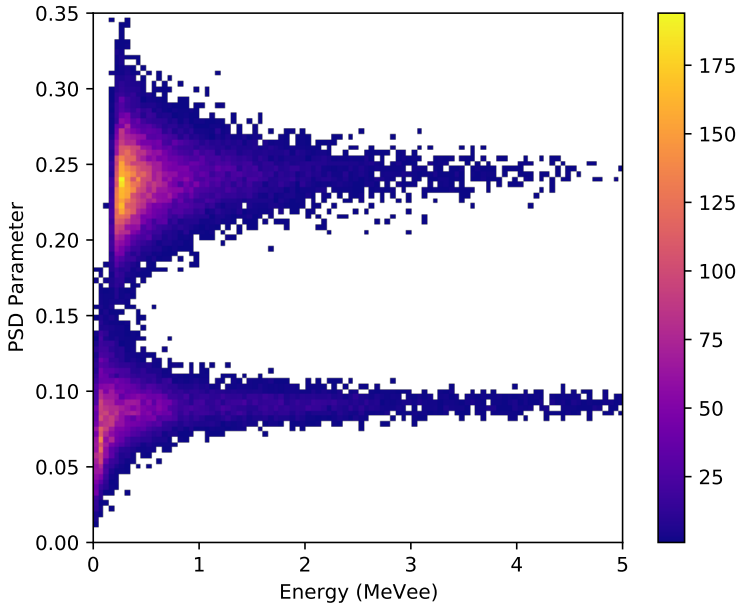

Figure 6.3: Histogram demonstrating how `DRiFT` handles a higher digitizer trigger threshold. Notice how low-energy neutrons are preferentially cut.

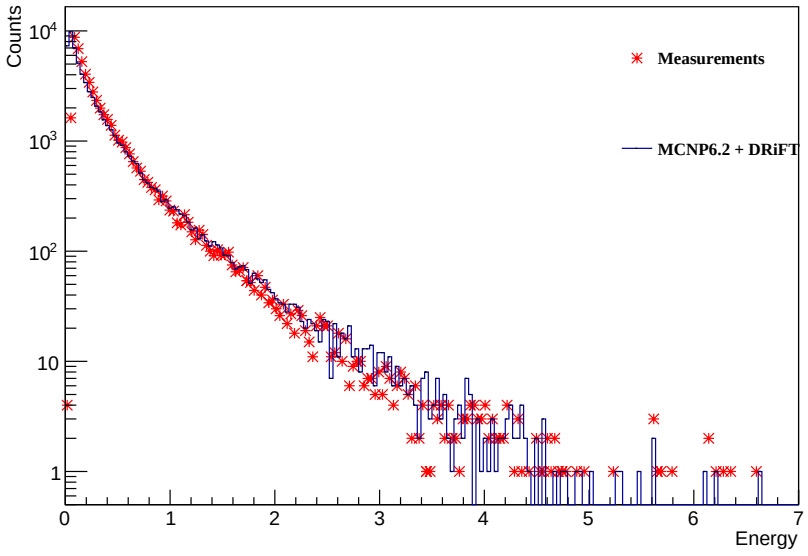

Figure 6.4: Digitizer processing with correct PMT voltage

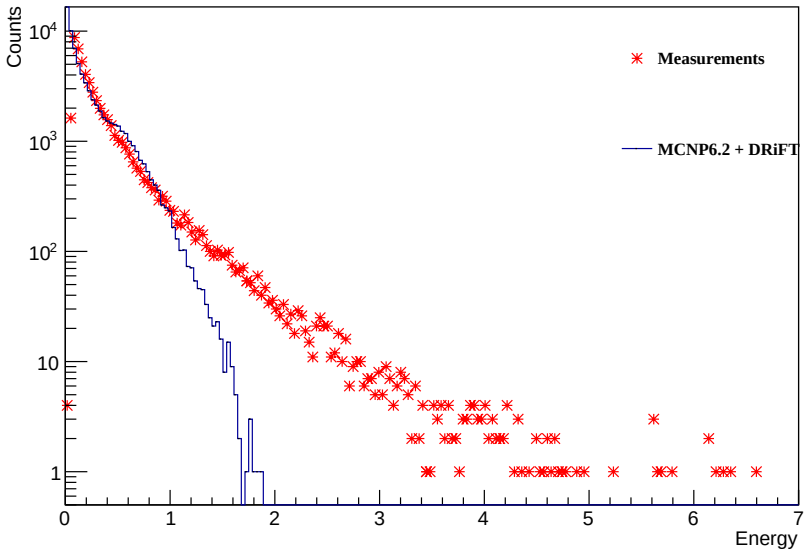

Figure 6.5: Digitizer processing with an incorrect applied PMT voltage

Part III

Additional DRiFT Features

Chapter 7

Source Particle Information in DRiFT

7.1 Overview

In many applications it is useful to know information about the original source particle that created a detector response. In these scenarios `DRiFT` can be used to print out information about the original source particle in the post-processed MCNP simulation. Source particle information can be written out using the `WriteOutput`, module described in the next chapter. Source information output by `DRiFT` includes a source particle's type, time, energy, and cell.

7.2 Use in DRiFT input file

The `SourceInformation` call is applied before any detector response is simulated.

```
[SourceInformation]
call=SourceInformation
```

If this call is present in the `DRiFT` configuration file and no source history is found in the corresponding `PTRAC` file, `DRiFT` will terminate with the following message:

```
terminate called after throwing an
instance of 'std::invalid_argument'
  what():  no source particle information
     found in PTRAC file
```

This usually happens when the `SRC` is not specified in the `MCNP` PTRAC card.

7.3 Keywords

7.3.1 Multiple Source particles: `multi_src`

Currently when emitting multiple fission neutrons MCNP will record the first emission in a history as a SRC event in the PTRAC card. Subsequent events are recorded as banked (BNK) interactions in the PTRAC card.

DRiFT identifies source particles based on the timestamp of the events in the PTRAC file. This is done by identifying the events with timestamp of zero because MCNP assigns a timestamp of zero to the first event of every history. This implementation enables DRiFT to identify sources particles both in the case of non-multiplying and multiplying sources. In the latter case, by setting `multi_src = yes`, multiple source particles are identified for each history, for example, from simulations emission of multiple neutrons and photons. This is important when using, for example, FREYA to simulate fissioning sources.

Default

By default this option is turned off.

7.3.2 Multiple Source particles, list of source particle: `source_particles_list`

If `multi_src = yes`, the user has the option to list a number of source particles to track. This is done using the keyword `source_particles_list`, which takes integer valued particle designators based on MCNP's convention. For example, `source_particles_list=1 2` is used to identify neutron and photon source particles, respectively, attributed to each detection event. This feature can be used to generated detector pulses from neutron and gamma-ray detection with true PSD labels assigned to them. When providing a list of source particles of interest using the keyword `source_particles_list`, if none of the values provided correspond to the particle designators used by MCNP, the DRiFT run is terminated generating the following error message:

```
The list of source particles of interest
(source_particles_list) must belong to list of particles
tracked in MCNP. See Table 2-2 of the MCNP manual.
```

Default

By default `source_particle_list=1` (when `multi_src = yes`) for tracking neutron source particles.

7.3.3 Multiple Source particles, more information: print_src_particle_events and print_src_for_each_det

In addition, when multi_src = yes and list of source particles of interest are provided using the keyword source_particles_list, the user also has option to printout the information about the source events identified in the log file. This is done using the keywords print_src_particle_events and print_src_for_each_det. If print_src_particle_events=yes, for each source particle identified, the following information is printed out:

```
Current source:  <source particle type identified by DRiFT>;
from nps:  <NPS of current history>

Current source event cell:  <current event cell>; current value
of source_cell:  <source cell identify by DRiFT>

Current source event particle type:  <source particle type
from PTRAC>; Time:  <event timestamp from PTRAC>

Current source event:  <event type>

Current source event energy in MeV: <energy of source particle>
```

If print_src_for_each_det=yes, for each detection of electron or proton, the source particle attributed to the detection event is printed out as follows:

```
Electron detection; Source particle assigned:  <source particle
identified by DRiFT>

Proton detection; Source particle assigned:  <source particle
identified by DRiFT>
```

Default

By default these two options are turned off.

7.4 Default Example

```
[SourceInformation]
call=SourceInformation
multi_src=no
source_particles_list=1 (default value applied when multi_src=yes)
print_src_particle_events=no
print_src_for_each_det=no
```

Chapter 8

Distributing Source Particles in Time

8.1 Overview

As pile up and dead time features are added to DRiFT the timestamp of the pulse will become significant. To date all simulations with MCNP + DRiFT have used a fixed source with an emission time of zero. The time distribution option was created to distribute the source particle emission times according to an activity defined by the user. DRiFT will assign the particle histories a new timestamp calculated using the activity value provided by the user. The time difference between the current event and the previous event, time_from_prev (second) will be calculated and used in future pile up calculations.

8.2 Use in DRiFT input file

The TimeDistribution call is applied before any detector response is simulated. Many detector response functions will convert the particle's original energy to 0.

```
call=TimeDistribution
```

If this call is present in the DRiFT configuration file and no source history is found in the corresponding PTRAC file, DRiFT will terminate with the following message:

```
terminate called after throwing an
instance of 'std::invalid_argument'
   what():  TimeDistribution only works
      with source information
```

This usually happens when the SRC is not specified in the MCNP PTRAC card.

8.3 Limitations

Use of this feature requires the user provide a PTRAC file with at least 100 * activity source histories. If DRiFT does not find a sufficient number of histories to properly distribute the particles in time, it will throw a invalid_argument error and exit.

8.4 Keywords

8.4.1 activity

Source activity is specified in Bq by the user and its use is mandatory if the TimeDistribution call is used.

8.5 Configuration Input Example

```
[TimeDistribution]
call=TimesDistribution
activity = 10
```

Chapter 9

Text File Output

9.1 Overview

To create text file outputs when executing DRiFT the user must call `WriteOutput`. This will create a text output file containing information specified in the keyword options described below. Each event (or energy deposition in the detector cells) is in a seperate row even if they originate from the same source particle. If the user reqeusts NPS information for each event, using the `nps` option of the `outputs` keyword (see next section), multipe events originating from the same source particle are simply identified by examining the NPS column of the output file.

9.2 Write Output Keywords

9.2.1 Output File Name, `output`

This option may be used to name the text output file that is created.

```
output = output_name.txt
```

If no output filename is selected, DRiFT will use the name of the configuration file (minus any extensions) to create a text output file. For example if the DRiFT input file was `EJ301.ini` the `WriteOutput` output created will be named `oEJ301.txt`.

9.2.2 `header`

If you wish to have a header in your file that contains names for each column, set `header=yes`.

9.2.3 Outputs

To retrieve the information you want, simply write `outputs=option A... option B ...`. The options available to write out include: `corr_count` which will print 'no'

for uncorrelated events, 'double' or 'triple' for two or three events, respectively, originating from the same source particle, `cells_history` which will print the history of all cells a particle interacted with before being detected, `det_pulse` which will print the converted energy of the detected event, `det_e` which will print the energy deposition of the event, `det_c` the charge deposition, `source_e` the source particle energy, `source_t` the source particle time, `source_cell` the origin cell of the source particle, `source_type` the source particle type, `PSD` a calculated PSD value, `det_cell` the cell is which the event was detected, `nps` the MCNP NPS number associated with that event (same output as specifying `count` in DRiFT intput), `time` the time of the detected event, `pileup` will flag pileup events, and `capture_loc` will print the X,Y,Z coordinates of the detection event.

9.2.4 `cell_flags`

You can track which detector events had any interaction with a specific cell (or cells) by setting `cell_flags=cells you are interested in` seperated by spaces. For example this is useful if you're looking at room return contributions to a signal.

9.2.5 Writing out all events in simulation `all_events`

If the user wishes to write out all events from a simulation (even those that did not result in a detector response) set `all_events=yes`, this option is set to no by default. Note, MCNP PTRAC files must contain source information for this feature to work.

9.2.6 CSV output

If you wish to have comma seperated output, selected `csv=yes`. Otherwise `DRiFT` will place just spaces between different outputs.

9.2.7 e_conversion

If you wish to convert your energy output into another, you can have it multiplied by a factor defined in the input, by default `e_conversion=1`.

9.2.8 corr_only

To write out ONLY correlated events, set `corr_only=yes`.

9.2.9 Pulse Shape Output File Name, `pulse_shape_output`

To write out the pulse shapes generated from each detection event, provide the file name using `pulse_shape_output=filename`. The detector pulse shapes are written out to this separate file, which is saved in the same directory that contains the main output file. This separate output file is only generated if the file name is provided using the `pulse_shape_output` keyword.

Each row of the pulse shape output file corresponds to a single detector pulse. The i^{th} value of each row shows the amplitude value of the i^{th} data point of the pulse. The number of pulse data points (or the length of each row) is determined by the value provided for the `digitizer_samples` keyword when calling the [Digitizer] module. The default value of `digitizer_samples` is 512. If a different value is not provided, each pulses (or each row of the pulse output file) will have 512 points. The time width between the pulse data points is determined by the value provided for the `rate` keyword when calling the [Digitizer] module. The `rate` value corresponds to the digizer's sampling frequency. As a result, the time difference between each pulse data point is equal to 1/`rate`. Since by default, `rate=500e6` (corresponding to sampling frequency of 500 MHz), if a different value is not provided, the pulse data points will be 2 ns apart. The amplitude values are written in terms of digitizer unit. As a result, the range of the amplitude values is determined by the value provided for the `resolution` keyword when calling the [Digitizer] module, which by default is 16384 (corresponding to 14 bit digitizer). The amplitude in terms of digitizer units is in turn a renormalized value showing how the amplitude values in Volts compare with the digitizer's dynamic range, which is given by the `voltage_range` keyword used by the [Digitizer] module.

Part IV

Test Suites and Examples

Chapter 10

Neutrons on EJ-301 Test Suite

10.1 Overview

This test case is found in the folder test_suite/Cf252_EJ301. This directory contains i) an MCNP input deck, *mcnp*, ii) measurement data in an ASCII format, *measurements*, iii) example DRiFT configuration files that output ROOT and/or ASCII format outputs, *drift.ini*, *drift_rootOnly.ini*, and *drift_asciiOnly.ini*, and iv) sample ROOT and Python analysis codes, *rootAnalysis.C* and *pyDrift.py*, Note, however, that references to ROOT software are not applicable to the executable release at this time. In addition to these files is the directory *standard_outputs_06_29_21*, which contains saved outputs from MCNP + DRiFT that can be used as a comparison for newly generated output files to check consistency. The included scripts *rootCheck.C* and *pyCheck.py* will plot the archived outputs against any newly generated output and the measurement data. Finally, the directory contains a readme that gives an overview of the test case and how to run it.

To confirm DRiFT is functioning properly, execute the following steps.

To run the case step-by-step, follow the below process:

- Source the environment file provided in DRiFT:

  ```
  source ../../load_modules.sh
  ```

  Ensure that your DRIFTDATA variable is properly set, executing the command below if needed:

  ```
  export DRIFTDATA=/PATH/TO/DRIFT/data
  ```

- Execute MCNP with *mcnp* as the input and *omcnp_* as the output:

  ```
  mcnp6 i=mcnp n=omcnp_
  ```

- Execute the DRiFT configuration file (note that if you wish to create only an ASCII or only a ROOT output file, replace *drift.ini* with *drift_asciiOnly.ini* or *drift_rootOnly.ini*, respectively):

```
../../build/drift drift.ini
```

- Run the analysis script. The python script reads the ASCII output format while the ROOT script reads the ROOT output format:

  ```
  root rootAnalysis.C
  ```

  ```
  python pyDrift.py
  ```

  The scripts are designed to accept command line arguments as follows:

  ```
  root 'rootAnalysis.C
     ("InputFilename.root","MeasurementFilename","OutputFilename")'
  ```

  ```
  python pyDrift.py -i InputFilename -m MeasurementFilename
     -o OutputFilename
  ```

  You should now see a plot comparing MCNP + DRiFT output with measurements, as shown in Figure 10.1.

- Run the consistency check scripts. Again, the python script reads the ASCII format and the ROOT script reads the ROOT format:

  ```
  root rootCheck.C
  ```

  ```
  python pyCheck.py
  ```

  You should now see a plot comparing your MCNP + DRiFT output with both the measurement and historical output, as shown in Figure 10.1.

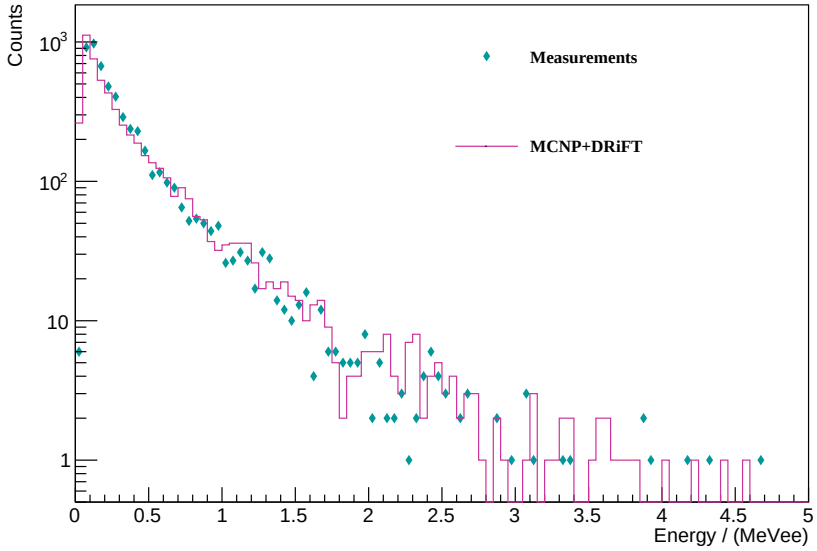

Figure 10.1: Comparing MCNP PTRAC data post-processed by DRiFT with Measurements.

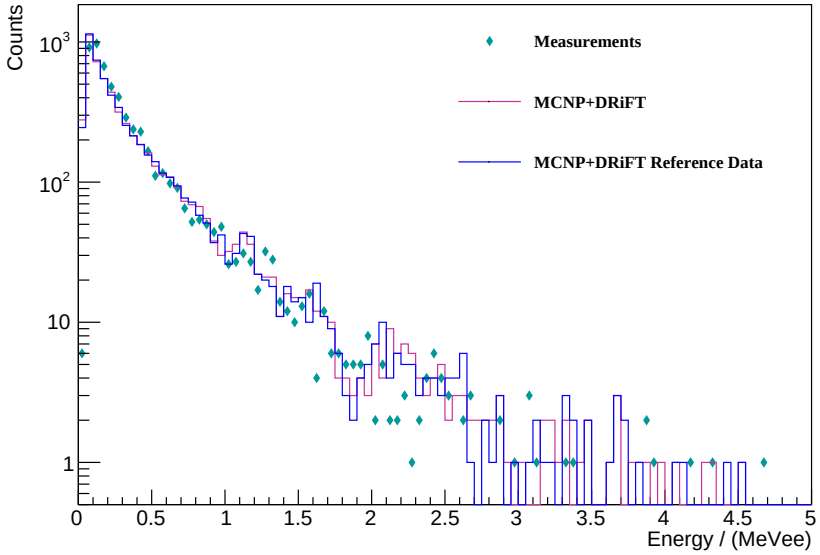

Figure 10.2: An example comparison of new and archived ROOT + DRiFT data.

10.2 List of Files in Cf252neutron test_suite directory

- **readme** - A README that contains useful information for running the test case.
- **rootAnalysis.C** - A ROOT macro which when executed will produce a plot of measurements and MCNP + DRiFT output. The script reads a .root DRiFT output file.
- **pyDrift.py** - A Python script which when executed will produce a plot of measurements and MCNP + DRiFT output. The script reads a .txt DRiFT output file.
- **mcnp** - MCNP6.2 input deck which will produce the required ptrac file: *omcnp_p*
- **drift.ini** - DRiFT configuration file which will create an output to be compared to measurements. The parameters defined in this file shapes how DRiFT will post-process the MCNP PTRAC file. Produces both a .root and a .txt output.
- **drift_rootOnly.ini** - DRiFT configuration file which will create an output to be compared to measurements. Produces only a .root output.
- **drift_asciiOnly.ini** - DRiFT configuration file which will create an output to be compared to measurements. Produces only a .txt output.
- **measurements** - Post-processed Cf252 data, with energy (column 1) and psd value (column 2).
- **rootCheck.C** - A ROOT macro which when executed will produce a plot of measurements, MCNP + DRiFT output, and archived MCNP + DRiFT output for comparison. The script reads a .root DRiFT output file.
- **pyCheck.py** - A Python script which when executed will produce a plot of measurements, MCNP + DRiFT output, and archived MCNP + DRiFT output for comparison. The script reads a .txt DRiFT output file.
- **standard_outputs_06_29_21** - A directory containing:
  - **drift.root** - An archived ROOT format output file, used by rootCheck.C.
  - **output.txt** - An archived ASCII format output file, used by pyCheck.py.
  - **example_slurm_output** - An archived output from the submission of a batch file that shows the process of running DRiFT in batch mode.
  - **python_output.pdf** - An archived image of the output of the Python analysis script.
  - **root_output.pdf** - An archived image of the output of the ROOT anlysis script.

10.3 Sample MCNP Input Deck

```
Creating PTRAC output for neutrons from Cf-252, An Example
1 1000  -0.874  -99     imp:n,h=1
2 2000  -1.2e-3 -98 99  imp:n=1 imp:h=1
3 0             98      imp:n,h=0

99  rcc  0 0 0  0 0 7.1  3.1
98  so   20

nps 5e5
mode n h
PHYS:N 50 50 4j 1
PHYS:h 50 50 -1 j 0 j 0 j j j 1 0 0 0.917
cut:n j 0 0 0
cut:h j 0 0 0
c This PTRAC File is what DRiFT actually reads.  It must be
c in BINARY format.
PTRAC File=BIN MAX=1e9 WRITE=ALL type=h,n event=bnk,src
sdef PAR = n erg = d1 pos = 0 0 -7
c ------------------------------
c Scintillator EJ-301
c ------------------------------
m1000  1001 0.5477
       6000 0.4523
c ------------------------------
c Air
c ------------------------------
m2000  7014 -0.75
       8016 -0.25
c Source particle energies distributed according to Cf-252
c Watt Spectrum
SP1 -3 1.18 1.03419
c Energy Deposition tally in cell 1, dose factors designated
below F6:h 1
DE6 lin 0    0.1  0.2  0.3  0.4
        0.5  0.6  0.7  0.8
        0.9  1.0  1.5  2.0  2.5
        3.0  3.5  4.0  4.5  5
        5.5  6    7    8    9    10
DF6 lin 0       0.1804  0.1802  0.2196  0.249
        0.2954  0.2954  0.3143  0.3314
        0.347   0.4265  0.4265  0.5095  0.5095
        0.5095  0.566   0.566   0.610   0.643
        0.643   0.643   0.671   0.694   0.714   0.731
F8:h 1 $Pulse height distribution tally in cell 1
```

```
FT8 PHL 1 6 1 0
E8:  0 9i 10 $Energy Bins for pulse height tally
F18:h 1 $Pulse height distribution tally in cell 1
E18:  0 9i 10 $Energy bins for pulse height tally
print 110
```

10.4 Sample DRiFT Parameter File

```
[global]
modeltype=event
datasource=mcnp
ptrac_type=bin
#Name of the PTRAC file you want to process
datafile=omcnp_p
#datafile is the file name of the mcnp ptrac output
det_cells=1

[SourceInformation]
call=SourceInformation
multi_src=yes

[Scintillation]
call=Scintillation
detector=EJ301
optical_transport=0.6
pmt_type=9821B
voltage=1500
divider_option=B

[Digitizer]
call=Digitizer
voltage_range =2.0
digitizer_samples=256
resolution=16384
ter_res = 50
DC_offset = 0.1
start_point = 0.1
digitizer_rate=500.e6
trigger_ADC=70
PSD=yes

[WriteOutput]
call=WriteOutput
outputs=source_e count det_pulse det_cell corr_count time PSD
cells_history
output=output.txt

[WriteHistogram]
call=WriteEventHistogram
bins=100
max_time=100
xmin = 0
xmax = 5
```

```
energy_units = MeVee
filename = drift.root
```

Chapter 11

Other Test Cases

11.1 Overview

In addition to the Cf-252 Neutrons on EJ-301 test case described above, the test_suite directory contains a number of other test cases to explore the functions and capabilities of DRiFT. The sections below give a brief description of each case. Note, that references to `ROOT` software are not applicable to the executable release at this time.

11.2 Activity

This test case simulates 14 MeV neutrons on a sphere of EJ-301. This test case does not contain any measurement data. Instead, the sample ROOT and Python analysis scripts, *rootAnalysis.C* and *pyDrift.py*, plot only the output of MCNP + DRiFT, as seen in Figure 11.2. The test case also includes archived outputs for consistency checks, contained in *standard_outputs_06_30_21*, and scripts that plot both a new MCNP + DRiFT output and the archived results, *rootCheck.C* and *pyCheck.py*. The general process for running this test case is the same as for the Cf-252 EJ-301 test case, with the exception that the measurement file is omitted as it does not exist. The default settings of the analysis scripts have been modified so that no parameters need to be specified if you use the default settings as defined in *drift.ini*.

11.3 Na22_EJ301

This test case simulates a Na-22 point source positioned near a cylinder of EJ-301. This test case contains measurement data for comparison, saved in *measurements*. Archived outputs are contained in the directory *standard_outputs_06_29_21*, which can be compared against any new MCNP + DRiFT output using the *rootCheck.C* or the *pyCheck.py* scripts. The process for running this test case is identical to that of the Cf252_EJ301 case.

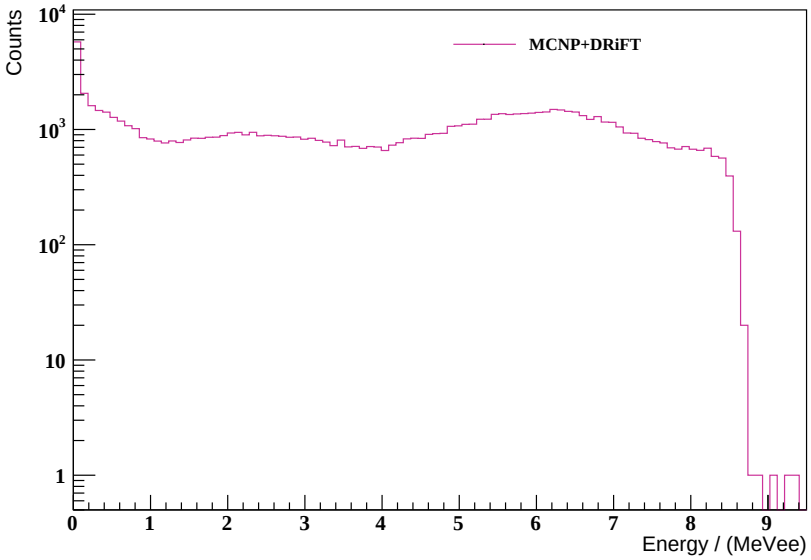

Figure 11.1: The MCNP + DRiFT output of the Activity test case.

An example output from this case is given in Figure 11.3.

11.4 PMT

This test case simulates the same geometry as in the Cf252_EJ301 test case. However, the applied voltage of the PMT, as defined in input parameter files, is set to 1200V, 1500V, and 2000V. This is to demonstrate the effects that applying an incorrect voltage to a PMT can have on the detector response. For the 1200V case, the gain is insufficient to produce a substantial enough electronic signal to trigger the analog to digital converter to record an event. This means that all except for the highest energy events which will naturally produce more light and thus a greater signal are lost. For the 2000V case, the pulses generated by the PMT are clipped by the 2V dynamic range of the digitizer. This leads to the distorted specturm at higher energies where the pulses are largest. The 1500V case represents the correct voltage to be applied to the PMT. These effects can be seen in Figure 11.4.

11.5 Using_Data_Tables

This test case again simulates the same geometry as in the Cf252_EJ301 test case. The purpose of this case is to demonstrate the ability of the user to define their own scintillators and PMTs without needing to modify the DRiFT code or recompile DRiFT. This is

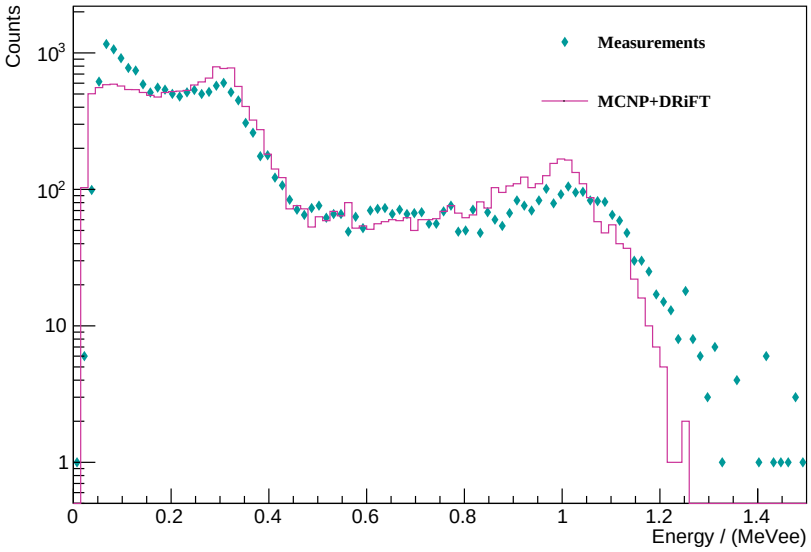

Figure 11.2: The MCNP + DRiFT output of the Na22_EJ301 test case, plotted alongside measurement data.

done by providing data files in the DRiFT input file, *drift.ini*. The scintillator emission spectrume (*PE_file*), the scintillator light output (*light_file*), and the PMT quantum efficiency (*QE_file*) must all be provided, along with parameters for the scintillation yield of the scintillator (*scint_yield*) and the gain of the PMT (*gain*). The user is referred to Chapter 5 for more information on this test case, and for more information on how to model new scintillators using data files.

11.6 Deuterated_Scintillator

This test case simulates the same geometry as in the Cf252_EJ301 case, but performs the simulation twice: once for a normal EJ-301 detector and once for a deuterated EJ-301D detector (i.e. the mass 1 hydrogen in the detector is replaced with mass 2 deuterium). The example root macros and python scripts for this test case will plot the results from both scintillators against the measurement data for EJ-301. The results of these scripts can be seen in Figure 11.6. Note that the light output response is different for the two detector types, as recoiling deuterons have a different stopping power in the scintillator than recoiling protons. The resultant pulse shape is also different. Both of these factors, combined with the different behavior of deuterons when bombarded with neutrons, leads to the different spectra seen in Figure 11.6.

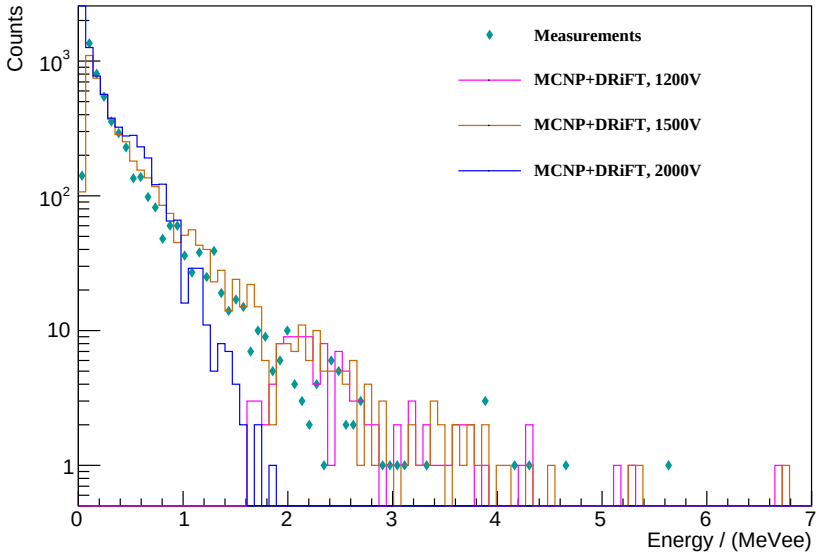

Figure 11.3: The MCNP + DRiFT output of the PMT test case, plotted alongside measurement data. Note the distorted spectra caused by applying the incorrect PMT voltages.

11.7 Scintillator_Comparison

This test case simulates a cylindrical scintillator detector similar to the Cf252_EJ301 test case. However, in this case, the detector is filled with 3 different types of scintillator: EJ-301, EJ-228, and EJ-200. In each case, a point source of 14 MeV neutrons is placed near the detector to create a response. The response of each scintillator, as output by DRiFT, can then be plotted in order to investigate the different responses of the three different kinds of scintillator. Note that one of the most significant differences between the three scintillators is in their light output response functions (the conversion from the energy of the recoil proton in the detector in MeV to the equivalent light output in MeVee). This results in the different endpoints for the three detectors, as seen in Figure 11.7.

11.8 Analysis of individual pulse shapes generated by DRiFT

This example demonstrates extraction of individual pulse shapes generated by DRiFT, pulse shape discrimination, and neutron-neutron doubles rate for differnt correlation angles, using Cf-252 source and five EJ-301 detectors. In particular, the correlated

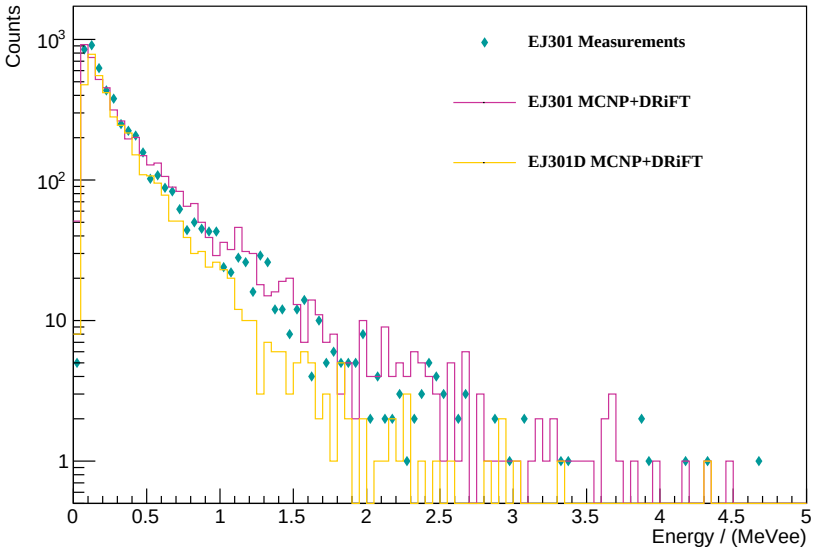

Figure 11.4: The MCNP + DRiFT output of the Deuterated_Scintillator test case, plotted alongside measurement data. Note the distinct spectra of EJ301 and EJ301D.

neutron measurment described in reference [9] is simulated to demonstrate DRiFT's source infomration features. The setup includes five EJ-301 detectors arranged along a semi-circle in increment of 45 degrees between the the detectors (see Figure 11.8). While the paper shows results for 20.4 cm and 50.4 cm radius setups, this example only simulates the latter.

Two separate MCNP simulation runs are perfomed in sequence using Cf-252 as a neutron source and Co-60 as a gamma-ray source, respectively, to allow DRiFT to generate neutron and gamma-ray detector pulses based on the PTRAC outputs created at the end of the MCNP runs. Future updated version of DRiFT will allow simultanious extraction of neutron and gamma-ray pulses from a single MCNP run using sources such as Cf-252. By specifying pulse_shape_output = user_defined_output_filename in (the WriteOutput section of) the DRiFT input file, as shown below,

```
    [WriteOutput]
call=WriteOutput
pulse_shape_output = user_defined_output_filename
outputs=source_e count det_pulse det_cell corr_count time PSD
cells_history
```

in addition to the default event summary output file, a separate file that containes the individual pulses corresponding to each detection event is generated for further post-processing.

The neutron and gamma-ray pulses generate are then post-processed using a python

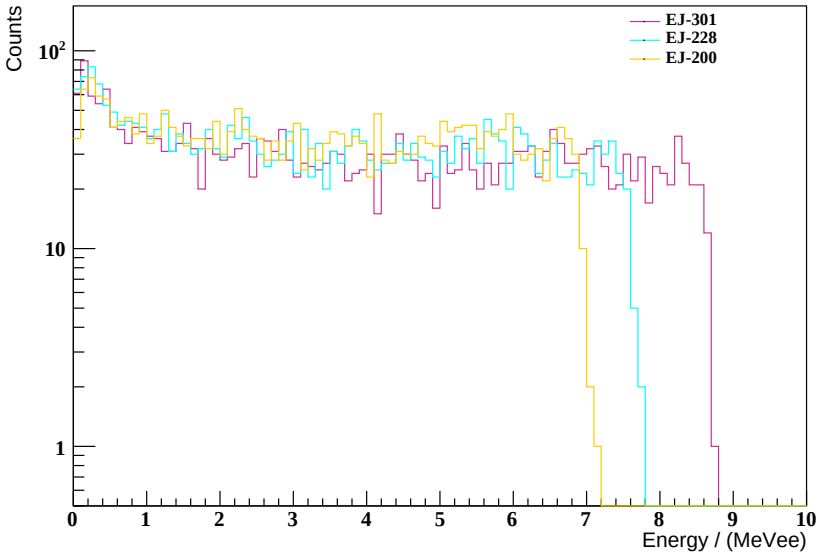

Figure 11.5: The MCNP + DRiFT output of the Scintillator_Comparison test case. Note the distinct endpoints of the three spectra.

script (Analyze_drift_pulses.py located in Cf252_EJ301_corr directory within test_suite) to perform pulse shape discriminations (PSD). Three different types of PSD methods are used to create 2D scatter plots of the PSD paramerers, each one showing the expected separation of neutron and gamma-ray pulses, based on the distinctive features of the two types of pulses in EJ-301 detectors, see Figure 11.8. This demonstrates the proper simulation of the detector response by DRiFT.

In the case of the Cf-252 source, simultanious emission of neutrons is simulated in MCNP using the FREYA option (fmult 98252 method=6). The events summary output file generated by DRiFT is then post-processed in the same python script (Analyze_drift_pulses.py) to generate a histogram of neutron-neutron doubles event for correlation angles of (45, 90, 135, 180) degrees. The histogram generated, Figure 11.8, shows higher number of doubles event occuring for the lower correlation angles, and vise-versa. While simulatanious emission of neutron emission from fission events are expected to lead to simultanious emission of neutrons mostly 180 degrees apart, the semi-circle arrangment of the detectors does not allow the capture of most of those neutrons since there is only one pair of detectors in 180 degree orientation, whereas there are multiple pairs of detectors separated by smaller correlation angles. Further, the higher fraction of cross-talk between detectors closer to each other, reference [9], also contributes to the higher number of neutron-neutron doubles event for the smaller correlation angles. Future work will show direct comparison of these results with measurement data.

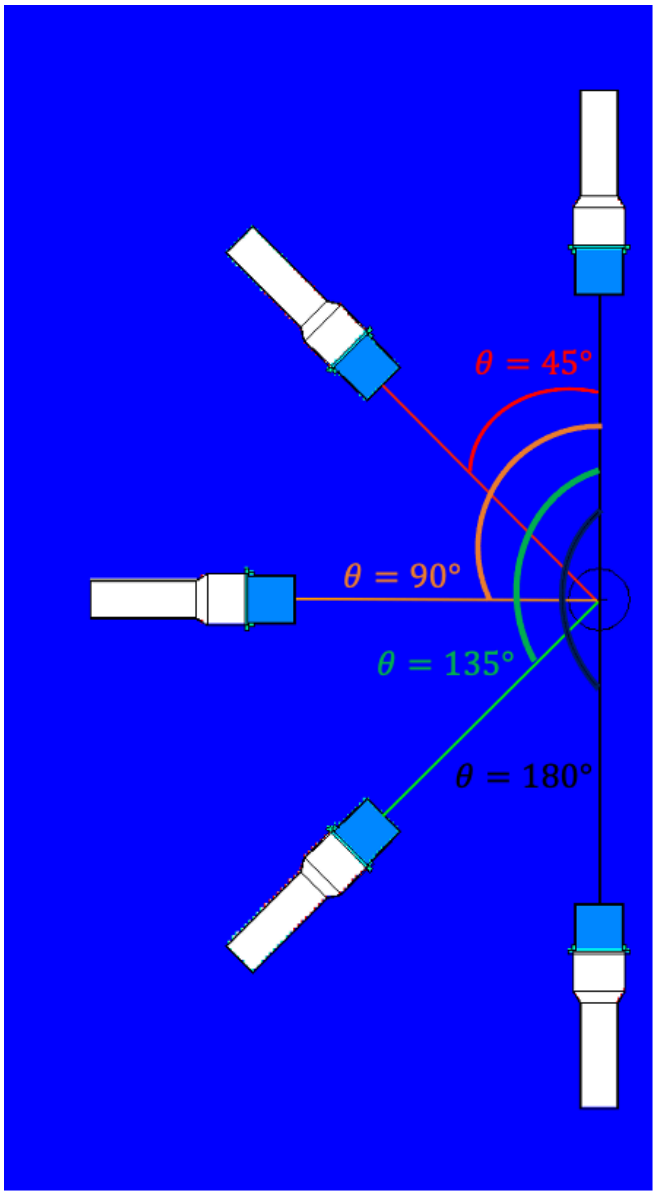

Figure 11.6: The schematic of detector setup used to extract correlated neutrons from Cf-252 source using EJ-301 scintillator detectors.

The slurm script included (slurm) runs, in sequence, the two MCNP simulations, followed by two drift runs post-processing the PTRAC outputs generated. And eventually, the python script is run leading to PSD and correlation analysis results saved in the same directory.

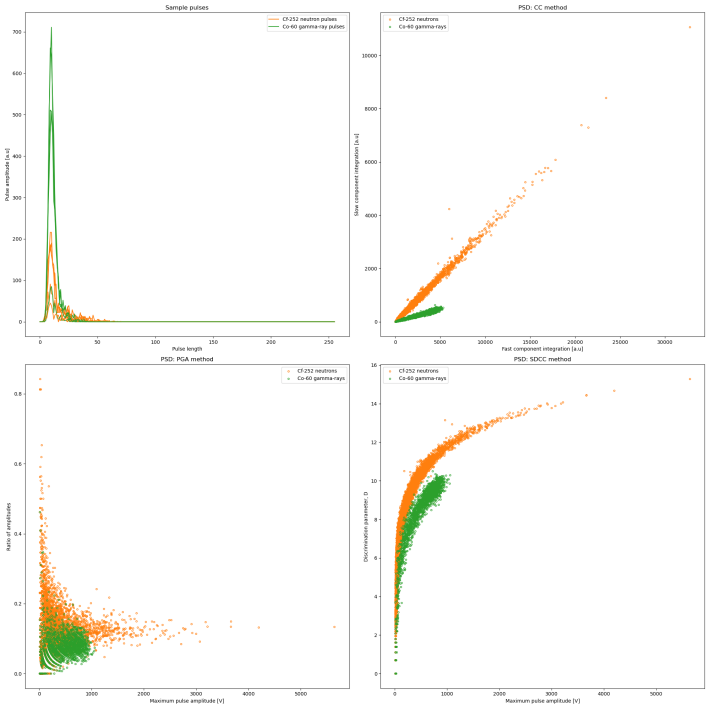

Figure 11.7: EJ-301 sample neutron and gamma-ray pulses generated by DRiFT, and the results of applying three different PSD methods on the pusles. Note that further optimization of the individual PSD method can improve the separateion between the two regions.

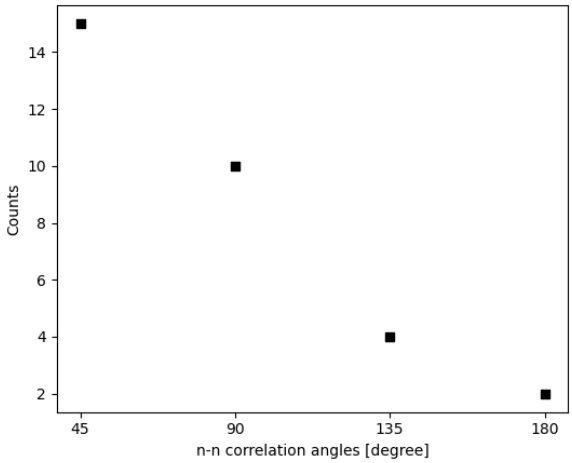

Figure 11.8: Histogram of neutron-neutron doubles event for a range of correlation angles between EJ-301 detectors.

Part V

Troubleshooting

Chapter 12

Common Execution Errors from Incorrect Input

12.1 General Input Errors

12.1.1 Forgetting to use the call keyword

The call keyword must be used for all non-global modules to be applied. If it is absent you will likely see the error terminate called after throwing an instance of std::bad_function_call.

For example:
[TimeDistribution]
call = TimeDistribution
activity = 100

12.1.2 Setting an incorrect DRIFTDATA environment variable

The environmental variable DRIFTDATA must be set to /path/to/drift/data. If the variable is pointing to the incorrect directory, you will likely see an error saying that a data file cannot be found (for example, the scintillator optical emission file below). You may also see a useful tip to check your DRIFTDATA setup.

```
   Cannot find file /this/directory/does/not/exist
     /scintillators/EJ301EmissionSpectrum check your DRIFTDATA
setup.
terminate called after throwing an instance of 'std::invalid_argument'
  what():  Scintillator optical emission file not found
```

12.1.3 Fix

Use the following command to properly set your DRIFTDATA variable:

```
  export DRIFTDATA=/path/to/drift/data
```

You can check that the directory is correct by entering:

```
ls $DRIFTDATA
```

If the command returns an error that the directory does not exist, re-export your DRIFTDATA variable, checking for any mistakes. If the problem persists, try navigating to the drift/data directory and entering:

```
export DRIFTDATA=$(pwd)
```

12.2 TimeDistribution

12.2.1 Activity too high for number of PTRAC histories

If the activity specified in TimeDistribution is high there may not be enough particle histories in the MCNP PTRAC file to properly distribute them in time. DRiFT will stop execution and diplay the error:

ERROR: Insufficient PTRAC histories for application of TimeDistribution.

*PTRAC file must contain at least [activity] * 100 histories for TimeDistribution to be used*

invalid_argument " Insufficient PTRAC histories for TimeDistribution

Fix

Run a longer MCNP simulation and store more PTRAC histories, or lower the source activity (if appropriate).

Bibliography

[1] John T Goorley, M James, T Booth, F Brown, J Bull, Lawrence J Cox, J Durkee, J Elson, M Fensin, RA Forster, et al. Initial MCNP6 Release Overview-MCNP6 Version 1.0. Technical Report LA-UR-13-22934, Los Alamos National Laboratory, 2013.

[2] Madison Theresa Andrews, Cameron Russell Bates, Edward Allen Mckigney, CJ Solomon, and Avneet Sood. Organic scintillator detector response simulations with drift. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 830:466–472, 2016.

[3] Madison Theresa Andrews, Amanda Christine Madden, Douglas R Mayo, and Surafel F Woldegiorgis. Prliminary helium-3 detector response capabilities in drift. Technical Report LA-UR-21-21199, Los Alamos National Laboratory, 2021.

[4] Madison Theresa Andrews, Surafel F Madden, Woldegiorgis, Douglas R Mayo, and Avneet Sood. Gas and semiconductor capability development in drift: Description, demonstration, and discussion. Technical Report LA-UR-21-25046, Los Alamos National Laboratory, 2021.

[5] Robert C Runkle, A Bernstein, and PE Vanier. Securing Special Nuclear Material: Recent Advances in Neutron Detection and their Role in Nonproliferation. *Journal of Applied Physics*, 108(11):111101, 2010.

[6] Sara A Pozzi, Marek Flaska, Andreas Enqvist, and Imre Pazsit. Monte Carlo and Analytical Models of Neutron Detection with Organic Scintillation Detectors. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 582(2):629–637, 2007.

[7] Madison Theresa Andrews, Cameron Russell Bates, Edward Allen Mckigney, CJ Solomon, and Avneet Sood. Development of a scintillator detector response post processing tool for mcnp output. *PHYSOR 2016*, 2016.

[8] Madison Theresa Andrews, Michael Evan Rising, K Meierbachtol, Patrick Talou, Avneet Sood, Cameron Russell Bates, Edward Allen Mckigney, and CJ Solomon. Characterizing scintillator detector response for correlated fission experiments

with mcnp and associated packages. *Radiation Physics and Chemistry*, 155:217–220, 2019.

[9] Matthew J Marcath, Madison T Andrews, Douglas R Mayo, Tyler A Jordan, and Michael E Rising. Experimental validation of scintillator detector response to correlated neutrons with mcnp and associated packages. *Radiation Physics and Chemistry*, 177:109131, 2020.

[10] Madison Andrews. Mcnp detector simulations with drift's digitizer class. Technical Report LA-UR-17-27860, Los Alamos National Laboratory, 2016.

[11] F.Q.L. Friesen and C.R. Howell. A functional form for liquid scintillator pulse shapes. *NIM A*, 2019.

[12] Madison Andrews. A EJ-301 neutron test case for DRiFT. Technical Report LA-UR-17-24266, Los Alamos National Laboratory, 2017.